Symbolic Constraints in Constructive Geometric Constraint Solving

Christoph M. Hoffmann*
Department of Computer Sciences
Purdue University

Robert Joan-Arinyo †
Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya

October 26, 1995

Abstract

In design and manufacturing applications, users of computer aided design systems want to define relationships between dimension variables, since such relationships express design intent very flexibly. This work reports on a technique developed to enhance a class of constructive geometric constraint solvers with the capability of managing functional relationships between dimension variables. The method is shown to be correct.

*Work supported in part by ONR Contract N00014-90-J-1599 and by NSF Grants CDA 92-23502, ECD 88-03017, and CCR 95-05745.

†While on leave in the Department of Computer Sciences, Purdue University. Partially supported by a CIRIT fellowship of the Government of Catalonia under grant 1995BEAI400071.

1 Introduction

In design and manufacturing applications, users of computer aided design systems are interested in defining functional relationships between dimension variables, since such relationships express design intent very flexibly. Some parametric relationships can be implemented by structuring the sketch appropriately, [3]. Moreover, simple functional relationships are the content of certain geometry theorems, such as the theorems of proportionality and many other classical results. Such theorems can be added to the constraint solver to extend its analysis capabilities. But in general, geometric constraint solving including functional relationships between dimension variables necessitates a more general approach and requires appropriate techniques and tools to achieve those functional capabilities that users expect.

This work reports on a technique we have developed to enhance constructive geometric constraint solvers with the capability of managing functional relationships between dimension variables. Essentially, we federate a purely geometric constraint solver with an equation solver. The communication between these subsystems is bi-directional.

Roughly speaking, we expect the geometric solver to have two phases, one that analyzes the constraint problem generically, devising a plan for constructing a solution, the second to carry out the construction with specific constraint values. We require that such a solver proceeds incrementally, and that the algorithm for deriving a construction plan be canonical in a sense made more precise later.

Similarly, we expect the equation solver to have two phases. The first phase reasons about the system of equations and isolates, on demand, a subset of equations that can be solved in principle. Here, the equation solver is told which variables are already valuated. In turn, the equation solver posts which other variables can be valuated by solving the isolated equations. In the second phase, the equation solver computes solutions of the various subsystems identified in the first phase. Note that such solvers can be implemented with minimal effort.

1.1 A Constraint Problem Example

We consider a two-dimensional geometric constraint problem in which there are explicit dimensions with numeric values and symbolic dimensions. We allow relationships between the symbolic dimensions given as a set of equations. Symbolic dimension values may be determined by computation, from the given equations, or by construction, from a partially completed geometric construction. There is no a-priori identification which symbolic dimensions are determined in one or the other way.

Figure 1 shows a kinematic problem from [5]. The problem is to design a crank-rocker linkage for a quick-return mechanism. The angular displacement

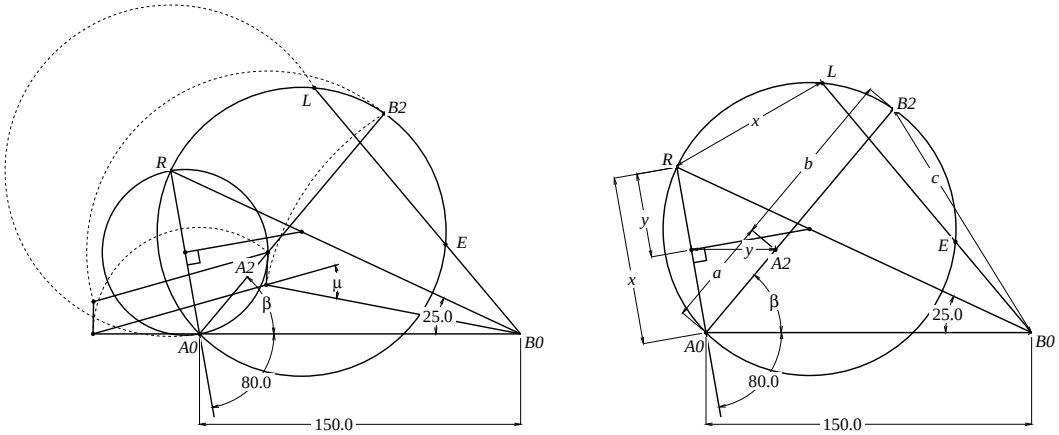

Figure 1: Example constraint problem: *left* formulation uses purely geometric constructions, *right* formulation uses symbolic dimensions and the equations $2y - x = 0$ and $b^2 + c^2 - (a + 150)^2 + 2bc\cos(\mu) = 0$ between them.

of the rocker is prescribed by $\psi = 50°$, the corresponding crank angle is of $\phi = 160°$, and the ground link length is $d = 150$. The object is to determine the link lengths and the minimum transmission angle μ. The left part of the figure expresses the construction of [5] purely geometrically, using circles and parallelograms to transfer equal measurements. The points L and E are the limit choices for the point $B2$ from which the link lengths can be determined as $a = \overline{(A0, A2)}$, $b = \overline{(A2, B2)}$, and $c = \overline{(B0, B2)}$. The particular choice of the point $B2$ could be driven by the angle β. The minimum transmission angle μ is then as shown. This angle ought to be maximized in a good design. Not only is this constraint problem difficult to understand, it is also difficult to solve and most variational solvers cannot find a solution.

Consider now the same problem using a solver that permits symbolic dimensions and equations between them. A formulation is shown on the right and the equations to be added are

$$\begin{aligned} 2y - x &= 0 \\ b^2 + c^2 - (a + 150)^2 + 2bc\cos(\mu) &= 0 \end{aligned}$$

To solve this problem, our solver alternates between constructing geometry and evaluating equations, as required by the nature of the problem. In the example, the construction begins by laying out the ground link and constructing the point R. Next, the symbolic dimension x can be determined from the geometry, and from it, by computation, y. Now the geometric construction continues with the circle through $A0$ and R, followed by the determination of the limits L and E. Then, after the user chooses a value for β, the values of the link lengths a, b

and c can be measured. From those measurements, the value of the minimum transmission angle μ is computed with the second equation.

Note that the second construction is much easier to comprehend. Furthermore, the required geometric constructions are a great deal simpler than in case of the purely geometric formulation.

2 Solution Method

First, we review the relevant aspects of constructive geometric constraint solvers and basic techniques for reasoning about systems of equations. Thereafter, we present the algorithm. To simplify the theoretical treatment, we will restrict to algebraic equations. Note that trigonometric functions can be expressed algebraically.

2.1 Constructive Geometric Constraint Solvers

Constructive geometric constraint solvers are based on the fact that most configurations in an engineering drawing are solvable by ruler, compass and protractor. In such solvers, the constraints are satisfied by incremental construction which makes the constraint process natural for the user and suitable for interactive debugging. The two main constructive approaches are *rule-* and *graph-constructive* approaches. Solvers based on a rule-constructive approach use rewrite rules for the discovery and execution of the construction steps. Graph-constructive solvers derive a sequence of construction steps by analyzing a graph representing the geometric constraints.

There is a general architecture for constructive geometric constraint solving systems that has been proved to be useful when all the constraints defined by the user are valuated; [2, 6]. This architecture splits the solution procedure into two main phases: the analysis phase and the construction phase.

In the analysis phase, a constraint graph is analyzed and a sequence of construction steps is produced. In the cited work, each step in the sequence corresponds to positioning three rigid geometric bodies that pairwise share a geometric element. We call this part the *g-analyzer*. In the construction phase the actual construction of the geometric elements is carried out according to the sequence determined by the analyzer. The construction is performed by solving certain standard sets of algebraic equations. We call this phase the *g-constructor*.

In the solver of [2], the geometric constraint problem is translated into a graph representing the constraints and geometric elements. The analyzer uses an associated rewriting system to generate specific geometric constructions. The constraint problem is solvable when the rewriting system has been reduced to a single element. Correctness of the analyzer has been shown in [4].

The solver of [6] generates a forest of trees where each tree in the forest represents a single constraint between two geometric elements. Here, the analyzer merges a small number of trees, and again each merging corresponds to a specific geometric construction. The problem is solvable if a single graph has been obtained. The correctness of the analyzer is established in [6].

2.2 Constraint Terminology

We consider geometric constraints from [2, 6]; that is, distance, angle, parallel, perpendicular, concentric, tangent, and so on. These constraints are extended by allowing symbolic constraints of distance and angle, where the "value" of the constraint is a variable symbol also called the *tag*.

A *valuated geometric constraint* is a distance or angle constraint whose value is a constant. Such a constraint is denoted $\mathbf{c}$ or $\mathbf{c}_i$.

A *symbolic geometric constraint* is a distance or angle constraint whose value is a variable tag. When the value of the variable can be determined, the constraint is converted into a valuated constraint. Symbolic constraints are denoted $\mathbf{c}^*$ or $\mathbf{c}_i^*$.

A *constraint equation* is an equation some of whose variables are tags of symbolic constraints. Depending on the power of the equation solver, there could be restrictions on the type of equations. For example, one could restrict to algebraic equations, or even to linear equations. In this paper we restrict to algebraic equations to simplify the theory of when a system of equations has a finite set of solutions.

A *geometric constraint problem* consists of a finite set of geometric elements g_k, valuated and symbolic constraints between pairs of geometric elements, and a set F of constraint equations.

The *geometric constraint graph* is a graph $G = (C, V)$, where V is the set of geometric elements of the problem, and C is the set of valuated geometric constraints. Associated with the graph is the set

$$\mathbf{T} = \{Z \mid Z \subset V\}$$

of *clusters*. The clusters are a set cover of V but not a disjoint set cover.

2.3 Constraint Equation Analysis

A *bigraph* $B = (E, F, X)$ consists of two finite sets F and X, the *vertices* of the graph, and a finite set of graph edges $E = \{(u, v) \mid u \in F, v \in X\}$. The edge $e = (u, v)$ is incident to the vertices u and v. We use bigraphs to analyze the structure of the equations relating the symbolic constraints. Here, F is the set of equations, X the set of all variables occurring in the equations. There is an edge (f, x), with $f \in F$ and $x \in X$ if the variable x occurs in the equation f. For an in-depth study of bigraphs see [7].

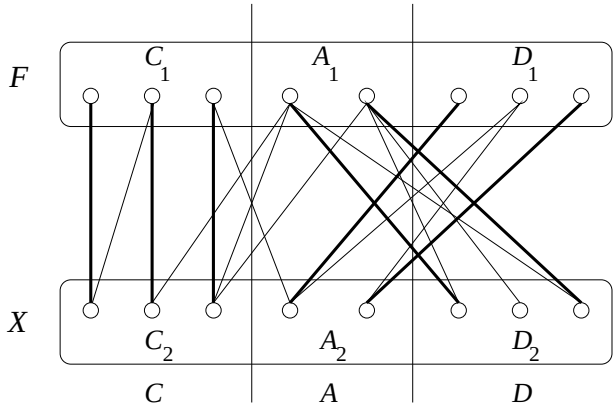

Figure 2: Canonical decomposition of a bipartite graph. A maximum matching is indicated by the heavy lines.

A subset M of edges in E is called a *matching* if no two edges in M are incident to the same vertex. A matching M is a *maximum matching* if $|M| \geq |M'|$ for every other matching M'. A vertex $v \in F \cup X$ is said to be *covered, matched* or *saturated* by a matching M if some edge of M is incident to v. Unmatched vertices are also called *unsaturated, uncovered* or *exposed*. A *perfect matching* is a matching which covers all the vertices of $V = F \cup X$.

If Y is any subset of $V = F \cup X$, let $\Gamma(Y) \subset V - Y$ denote all vertices that are adjacent to at least one vertex of Y. The *surplus* of Y is defined by $|\Gamma(Y)| - |Y|$. Intuitively, if the bigraph $B = (E, F, X)$ represents a system of independent algebraic equations, then the system is underdetermined if the surplus of X is negative, and is overdetermined if the surplus is positive.

Let $B = (E, F, X)$ be a bipartite graph, $V = F \cup X$. Denote by $D \subset V$ the set of all the vertices in B which are not covered by at least one maximum matching of B. Let A be the set of vertices in $V - D$ adjacent to at least one vertex in D. Finally let $C = V - A - D$. Furthermore, define the sets $A_1 = A \cap F$, $A_2 = A \cap X$, $C_1 = C \cap F$, $C_2 = C \cap X$, $D_1 = D \cap F$, $D_2 = D \cap X$. It is known, [7], that with these concepts, the bipartite graph B has a unique decomposition into three induced subgraphs, namely, $B_0 = (E_0, C_1, C_2)$, $B_1 = (E_1, A_1, D_2)$ and $B_2 = (E_2, D_1, A_2)$, plus additional edges between A_1 and A_2, A_1 and C_2, and between A_2 and C_1. The decomposition is called the *Dulmage-Mendelsohn* decomposition, hereafter called *DM-decomposition*. Figure 2 shows an example.

Consider the bigraph $B = (E, F, X)$ representing the given algebraic equations between the symbolic constraints. We assume that the equations are *algebraically independent.*[1] Any maximum matching of B will include a perfect matching of B_0 plus additional matchings in B_1 and B_2. The matchings of B_1

[1]Intuitively, no equation is redundant. For a precise definition see, e.g., [9].

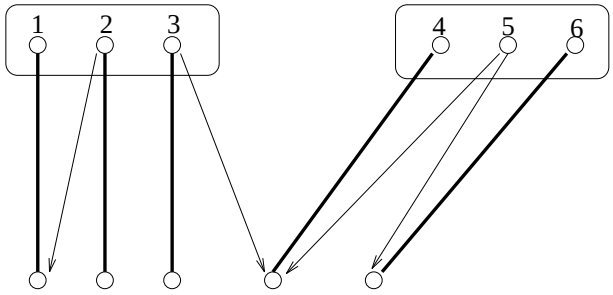

Figure 3: KH-decomposition of the solvable subsystem of Figure 2 shows that the equations (1), (4) and (6) can be solved separately. Equation (2) must be solved after Equation (1) and Equation (3) must be solved after Equation (4). Equation (5) is redundant and must be checked after Equations (4) and (6) have been solved.

and B_2 will cover the vertices in A_1 and A_2. Moreover, the DM-decomposition can be computed from a maximum matching in linear time; [1].

It is not difficult to see that the DM-decomposition permits isolating a set of equations that can be solved. The corresponding vertices are those of C_1 and the covered vertices of D_1. Since the matching covers all vertices of A_2, and since there cannot be edges between C_1 and D_2 and between D_1 and D_2, it follows that the set is well-formed and can be solved assuming all equations are independent. Note, however, that any uncovered vertices in D_1 correspond to redundant equations that must be evaluated for consistency.

The subsystem of solvable equations can be further decomposed by the *König-Hall decomposition*, abbreviated KH-decomposition. Consider the induced subgraph B_W of B corresponding to the solvable subsystem. Consider the edges of the matching bi-directional, and the remaining edges oriented from F to X. Let $W_1, W_2, ...$ be the strongly connected components of B_W with this orientation convention; [8]. Then the edges between the W_k induce an undirected, acyclic graph that expresses the dependencies between the subsystems corresponding to the components; [7]. A topological sort of the induced acyclic graph then gives the order in which to solve the subsystems corresponding to each strongly connected component. See also Figure 3.

2.4 Symbolic Constraint Analysis

The constructor finds a solution by executing a sequence of construction steps generated by the analyzer. All values needed for the construction must be available when the step is carried out. To satisfy this requirement, the analyzer classifies each symbolic constraint according to the way by which its values will be computed. We call a symbolic constraint *computable* when its value

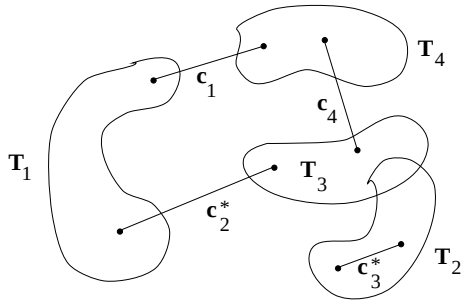

Figure 4: Situation where every further constructive step involves symbolic constraints.

is to be found by solving a subset of the constraint equations. We say that a symbolic constraint is *propagatable* when its value can be derived from geometric elements already placed with respect each other. When a constraint can be both computable and propagatable, it is considered to be propagatable by the analyzer.

Assume that, after having generated some construction steps, the analysis is not complete and the analyzer has reached a situation where no constructive rule applies because, in every possible new step, symbolic constraints are involved that are not yet valuated; see also Figure 4. We shall say then that the analyzer is in a *blocked* state. In order for the analyzer to proceed further, a subset of the symbolic constraints must be valuated. The analyzer will proceed as follows. First it will valuate all the propagatable constraints it can determine at that moment. Then the analyzer will request identification of a subset of constraint equations from which some computable constraints can be valuated. In general, several computable constraints have to be solved simultaneously.

2.4.1 Evaluating Propagatable Constraints

Let $\{\mathbf{B}_i\}$ be the set of geometric bodies already built, $\mathbf{T}_i$ the corresponding cluster consisting of the geometric elements of $\mathbf{B}_i$. Let $\{\mathbf{c}_j\}$ and $\{\mathbf{c}_k^*\}$ be the sets of valuated and symbolic constraints, respectively, that are defined on the geometric elements belonging to the $\{\mathbf{B}_i\}$; see Figure 4. Then a symbolic constraint $\mathbf{c}^*$ is propagatable at a given analysis step if, and only if, the geometric elements related by $\mathbf{c}^*$ belong to the same cluster $\mathbf{T}_i$. For example, the symbolic constraint $\mathbf{c}_3^*$ in Figure 4 is propagatable because it is defined between geometric elements already placed with respect each other in geometric body $\mathbf{B}_3$, whereas the symbolic constraint c_2^* is not propagatable.

When a blocked state is reached, the analyzer searches for all propagatable symbolic constraints. For each propagatable constraint with tag x, the following steps are done:

1. Derive the value of x from the constructed geometry.
2. Valuate the symbolic constraints labeled x.
3. Delete the vertex x and all incident edges in the bigraph.

Note that the last step reflects the fact that the value of variable x is now known. Since these steps do not define any geometric construction directly, we call them *computational steps*.

2.4.2 Evaluating Computable Constraints

Assume that the values of all propagatable symbolic constraints have been determined. We identify a subset of the constraint equations that is solvable, using the DM-decomposition and the KH-decomposition of the bipartite graph associated with the current set of constraint equations. By Section 2.3, the DM-decomposition is unique and determines a solvable subsystem. The further decomposition is for efficiency reasons. The computational steps are

1. Identify a solvable subsystem S from the bigraph B.
2. Solve S and check redundant equations for consistency.
3. For each computed variable x that is tag of a symbolic constraint, valuate the corresponding symbolic constraints in the constraint graph.
4. Remove the subgraph corresponding to S from the bigraph.

2.5 The Algorithm

The constraint solving algorithm is summarized below. We assume that the geometric solver uses a constraint graph for the analysis phase, and that the equation solver uses a bipartite graph for the constraint equations. Updating the knowledge data base summarizes the necessary action to post propagated or computed constraints to the geometric solver and the equation solver, as described before.

1. Compute the bipartite graph from the constraint equations;
   compute the constraint graph of the geometric problem.
2. `set` *blocked* to *true*.
3. `while` some construction rule can be applied `do`
   `set` *blocked* to *false*;
   generate the construction rule;
   update the constraint graph.
   `endwhile`
4. `if` a symbolic constraint can be propagated `then`

`set` *blocked* to *false*;
derive all propagatable constraint values;
update the knowledge data base;
delete the corresponding vertices and incident edges of B.
`endif`

5. Compute the DM-decomposition of the bigraph B;
identify a solvable subsystem S.

6. `if` there is a solvable subsystem S `then`
`set` *blocked* to *false*;
solve S;
update the knowledge data base;
delete the corresponding subgraph of B.
`endif`

7. `if` *blocked* `then` terminate;
`if` inconsistently overconstrained `then` terminate;
otherwise goto Step 2.

The loop in Steps (2–7) is terminated when Steps (3–6) cannot do any work or a potential inconsistency in the constraint equations has been found.

3 Correctness

We show that the algorithm is correct in the following sense. Let ρ denote a geometric reduction step, π the valuation of a propagatable constraint, and κ the evaluation of a computable step. We will prove the following statement:

> If there exists a sequence of steps of type ρ, κ and π that reduces the constraint graph to a single cluster and the bigraph to the empty graph, then the algorithm finds such a sequence.

Note that, a priori, there could be many different sequences, and that some of them could result in unsuccessful termination of the algorithm. To argue that this cannot happen, we need to introduce some concepts.

3.1 Geometric Constraint Analysis

In the correctness proof of the algorithm, we will use the terminology of [4]. There, a geometric constraint graph is structurally overconstrained if there is a vertex-induced subgraph with m vertices and more than $2m - 3$ edges. In particular, the constraint graph itself cannot be well-constrained if it has more than $2n - 3$ edges, where n is the number of vertices. We generalize these definitions now.

Intuitively, a constraint problem can be overconstrained in one of three ways: First, discounting the symbolic constraints, the constraint graph could be structurally overconstrained. Second, ignoring the geometric constraints, the system of equations supplied could contain an overdetermined subsystem. Third, the interaction of geometric and symbolic constraints introduces additional constraints, by valuating symbolic constraints, such that at some time an overconstrained, partially solved problem is obtained. In our definition, we only consider the first and the third possibility. Accounting thereafter for the second possibility is easy.

Definition 3.1
A geometric constraint problem is geometrically overconstrained if, for some derivable set of clusters $\mathbf{T}$, the associated geometric constraint graph $G = (C, V)$ is structurally overconstrained.

Definition 3.2
A geometric constraint problem is geometrically underconstrained if it is not geometrically overconstrained and for some derivable set of clusters, $\mathbf{T}$, the associated geometric constraint graph $G = (C, V)$ is structurally underconstrained.

Definition 3.3
A geometric constraint problem is geometrically well-constrained if for every set of clusters, $\mathbf{T}$, the associated geometric constraint graph $G = (C, V)$ is structurally well-constrained.

A geometric construction found by the g-analyzer induces a reduction ρ on the set of clusters associated with the geometric constraint graph. Specifically, a ρ-reduction merges three clusters $\mathbf{C}_1$, $\mathbf{C}_2$ and $\mathbf{C}_3$ that pairwise share a single geometric element. That is,

$$\begin{aligned} \mathbf{C}_1 \cap \mathbf{C}_2 &= \{g_1\} \\ \mathbf{C}_2 \cap \mathbf{C}_3 &= \{g_2\} \\ \mathbf{C}_3 \cap \mathbf{C}_1 &= \{g_3\} \end{aligned}$$

where the g_i are distinct and are not all lines. Let $\mathbf{T}_0$ be the initial set of clusters and let $\rightarrow^*$ be a possibly empty sequence of reduction steps. By [4], the rewrite system $(\mathbf{T}_0, \xrightarrow{\rho}{}^*)$ is terminating and Church-Rosser if the geometric constraint graph is not structurally overconstrained. In particular, two ρ reductions commute. Furthermore, if $\mathbf{T}_1$ is such that $\mathbf{T}_1 \xrightarrow{\rho} \mathbf{T}_2$ and $\mathbf{T}_1$ has a well-constrained associated geometric constraint graph, then the geometric constraint graph associated with $\mathbf{T}_2$ is also well-constrained.

3.2 Valuating Symbolic Constraints

Valuation by computation introduces a new valuated constraint in the geometric constraint graph. It also adds a new two-element cluster to the cluster set for every computed variable. This addition is called a κ-reduction. By convention, a κ-reduction computes a minimal independent solvable subsystem.

Valuation by propagation does not affect the geometric constraint graph or the cluster set, but will have an effect on the equation analysis because the symbolic constraint variable is no longer an unknown and, consequently, must be removed from the set of variables in the bigraph. We call this a π-reduction.

3.3 Unique Normal Form Property

The reduction system $(\mathbf{T}_0, \overset{\{\rho,\kappa,\pi\}}{\longrightarrow}{}^*)$ will be shown to be terminating. If the problem is not geometrically overconstrained, moreover, then we show that the reductions commute, i.e., that we have unique normal forms.

While termination is straightforward, unique normal forms must be argued both with regards to how reductions commute with each other and why those commuting properties justify Definitions 3.1–3.3.

Proposition 3.1

The reduction system $(\mathbf{T}_0, \overset{\{\rho,\kappa,\pi\}}{\longrightarrow}{}^*)$ is terminating.

Proof

Assume that the initial cluster set $\mathbf{T}_0$ has been obtained from a constraint graph with n nodes, where n_v edges correspond to the valuated constraints and n_s edges correspond to the symbolic constraints.

The κ and π reductions do not add new nodes to the constraint graph. Since each κ and each π reduction step reduces the number of symbolic constraints by at least one, the total number of such reductions is bounded by n_s. Each ρ reduction step reduces the number of clusters by 2. Each κ-reduction, moreover, adds as many 2-element clusters to the constraint graph as there are variables solved by the reduction. Thus, every reduction sequence has length less than $(n_v + n_s + 1)/2 + n_s$. □

Having established termination, we begin by showing that reductions of type ρ always commute with reductions of the other types.

Lemma 3.2

If $\mathbf{T}_1 \overset{\rho}{\rightarrow} \mathbf{T}_2$ and $\mathbf{T}_1 \overset{\pi}{\rightarrow} \mathbf{T}'_1$, then there is $\mathbf{T}'_2$ such that $\mathbf{T}'_1 \overset{\rho}{\rightarrow} \mathbf{T}'_2$ and $\mathbf{T}_2 \overset{\pi}{\rightarrow} \mathbf{T}'_2$.

Proof

A propagated symbolic constraint does not change the cluster set or the constraint graph. Hence $\mathbf{T}'_1 \overset{\rho}{\rightarrow} \mathbf{T}'_2$. Moreover, a ρ-reduction does not split a cluster, hence $\mathbf{T}_2 \overset{\pi}{\rightarrow} \mathbf{T}''_2$. Clearly $\mathbf{T}'_2 = \mathbf{T}''_2$. □

Lemma 3.3
If $\mathbf{T}_1 \xrightarrow{\rho} \mathbf{T}_2$ and $\mathbf{T}_1 \xrightarrow{\kappa} \mathbf{T}_1'$, then there is $\mathbf{T}_2'$ such that $\mathbf{T}_1' \xrightarrow{\rho} \mathbf{T}_2'$ and $\mathbf{T}_2 \xrightarrow{\kappa} \mathbf{T}_2'$.
Proof
The valuation of a symbolic constraint by κ-reduction adds a number of 2-element clusters and corresponding valuated constraints to the geometric constraint graph. Hence, using the same ρ-reduction, we have $\mathbf{T}_1' \xrightarrow{\rho} \mathbf{T}_2'$. Conversely, a ρ-reduction does not change the bigraph B, hence $\mathbf{T}_2 \xrightarrow{\kappa} \mathbf{T}_2''$.

States $\mathbf{T}_2'$ and $\mathbf{T}_2''$ clearly have the same constraint graph and the same set of clusters. Since the same variables have been computed, they also have the same bigraph, hence $\mathbf{T}_2' = \mathbf{T}_2''$. □

Since ρ-reductions commute with κ-reductions and with π-reductions, we can use this commutativity to justify Definition 3.1 in Corollary 3.4 and Lemma 3.5.

Corollary 3.4
Let $\mathbf{T}_1 \xrightarrow{\rho} \mathbf{T}_2 \xrightarrow{\kappa} \mathbf{T}_2'$ and $\mathbf{T}_1 \xrightarrow{\kappa} \mathbf{T}_1'$. Then $\mathbf{T}_1'$ is overconstrained iff $\mathbf{T}_2'$ is.
Proof
The statement is trivial if $\mathbf{T}_1$ has a structurally overconstrained constraint graph. By [4], if the constraint graph of $\mathbf{T}_1$ is not structurally overconstrained, then neither is the constraint graph of $\mathbf{T}_2$.

Assume that the constraint graph of $\mathbf{T}_1'$ is structurally overconstrained. Then there exists a nonempty set of equations $\{f_1, \ldots, f_m\}$ defining $m > 0$ valuated constraints which have been added by reduction κ to the constraint graph G_1 of $\mathbf{T}_1$. By Lemma 3.3, ρ and κ commute. Hence, $\{f_1, \ldots, f_m\}$ defines the same $m > 0$ valuated constraints in $\mathbf{T}_2$, yielding a structurally overconstrained constraint graph G_2' of $\mathbf{T}_2'$.

Now assume that the graph of $\mathbf{T}_2'$ is structurally overconstrained. This implies that reduction κ adds at least one valuated constraint to the constraint graph G_2 of $\mathbf{T}_2$ to give a structurally overconstrained graph G_2'. Since ρ and κ commute, the same set of valuated constraints can be added by reduction κ to the well-constrained graph G_1, resulting in an overconstrained graph G_1' associated with $\mathbf{T}_1'$. □

Lemma 3.5
Let $\mathbf{T}_0 \xrightarrow{\{\rho,\kappa,\pi\}*} \mathbf{T}_1$, $\mathbf{T}_1 \xrightarrow{\kappa} \mathbf{T}_2$ and $\mathbf{T}_1 \xrightarrow{\pi} \mathbf{T}_1'$. If the same symbolic constraint $\mathbf{c}^*$ of $\mathbf{T}_1$ is valuated by both κ and π, then the constraint problem $\mathbf{T}_0$ is geometrically overconstrained.
Proof
Since $\mathbf{c}^*$ is valuated by π, the valuated constraint is redundant to the geometric construction. But the tag x of $\mathbf{c}^*$ can be valuated by κ, so the constraint graph of $\mathbf{T}_2$ is structurally overconstrained. □

Lemma 3.6
Assume that $\mathbf{T}_1$ is not geometrically overconstrained. If $\mathbf{T}_1 \stackrel{\kappa}{\rightarrow} \mathbf{T}_2$ and $\mathbf{T}_1 \stackrel{\pi}{\rightarrow} \mathbf{T}'_1$, then there is $\mathbf{T}'_2$ such that $\mathbf{T}'_1 \stackrel{\kappa}{\rightarrow} \mathbf{T}'_2$ and $\mathbf{T}_2 \stackrel{\pi}{\rightarrow} \mathbf{T}'_2$.
Proof
By Lemma 3.5, the constraint variables valuated by κ and π must be disjoint. □

Lemma 3.7 ([4])
Assume that $\mathbf{T}_1$ is not geometrically overconstrained. If $\mathbf{T}_1 \stackrel{\rho_1}{\rightarrow} \mathbf{T}_2$ and $\mathbf{T}_1 \stackrel{\rho_2}{\rightarrow} \mathbf{T}'_1$, then there is $\mathbf{T}'_2$ such that $\mathbf{T}'_1 \stackrel{\rho_1}{\rightarrow} \mathbf{T}'_2$ and $\mathbf{T}_2 \stackrel{\rho_2}{\rightarrow} \mathbf{T}'_2$.
Proof
Since $\mathbf{T}_1$ is not geometrically overconstrained, the associated constraint graph is not structurally overconstrained. □

Lemma 3.8
Assume that $\mathbf{T}_1$ is not geometrically overconstrained. If $\mathbf{T}_1 \stackrel{\pi_1}{\rightarrow} \mathbf{T}_2$ and $\mathbf{T}_1 \stackrel{\pi_2}{\rightarrow} \mathbf{T}'_1$, then there is a $\mathbf{T}'_2$ such that $\mathbf{T}'_1 \stackrel{\pi_1}{\rightarrow} \mathbf{T}'_2$ and $\mathbf{T}_2 \stackrel{\pi_2}{\rightarrow} \mathbf{T}'_2$.
Proof
Trivial □

Lemma 3.9
If $\mathbf{T}_1 \stackrel{\kappa_1}{\rightarrow} \mathbf{T}_2$ and $\mathbf{T}_1 \stackrel{\kappa_2}{\rightarrow} \mathbf{T}'_1$, then there is a $\mathbf{T}'_2$ such that $\mathbf{T}'_1 \stackrel{\kappa_1}{\rightarrow} \mathbf{T}'_2$ and $\mathbf{T}_2 \stackrel{\kappa_2}{\rightarrow} \mathbf{T}'_2$.
Proof
If both κ_1 and κ_2 are applicable to $\mathbf{T}_1$, then the bigraph $B(\mathbf{T}_1)$ has a solvable subgraph that can be further partitioned into two disconnected subgraphs. Hence, there are two independent solvable sets of equations corresponding to the two reductions. □

These lemmas imply that, for problems that are not geometrically overconstrained, the rewrite system $(\mathbf{T}_0, \stackrel{\{\rho,\kappa,\pi\}}{\longrightarrow}{}^*)$ is terminating and has the Church-Rosser property. Therefore, it has a unique normal form. This establishes the correctness of our algorithm.

4 Summary

We have presented a technique for solving constraint problems that involve functional relationships between dimension variables. The method is especially effective for geometric constraint solvers that use the rewriting rule paradigm. A particular benefit of the approach is that the geometric solver can be, as it were, federated with a general purpose equation solver.

Our technique coordinates two sets of data, the geometric constraint data, and the symbolic equation data. Geometric data is represented by a set of

clusters. Symbolic equation data is represented by a bigraph. The information flow between these two structures is managed by two new rewriting rules: the κ-reduction rule and the π-reduction rule. The κ-reduction rule valuates symbolic constraints by solving a subset of constraint equations from the bigraph and adds the resulting constraints to the set of clusters. The π-reduction rule valuates symbolic variables in the set of equations by deriving them from already built geometry.

It has been shown that when the constraint problem is not overconstrained, the method is correct. That is, if there is a sequence of rewriting steps that reduces the constraint graph to a single cluster and the bigraph to the empty graph, then our method finds such a sequence. Thus, the simple-minded strategy of applying the rules in any convenient order is justified and we obtain an algorithm that is easy to implement.

References

[1] S. Ait-Aoudia, R. Jegou, and D. Michelucci. Reduction of constraint systems. In *Compugraphics*, pages 83–92, 1993.

[2] W. Bouma, I. Fudos, C. Hoffmann, J. Cai, and R. Paige. A geometric constraint solver. *Computer Aided Design*, 27:487–501, 1995.

[3] I. Fudos. Editable representations for 2D geometric design. Master's thesis, Purdue University, Dept. of Comp. Sci., 1993.

[4] I. Fudos and C. M. Hoffmann. Correctness proof of a geometric constraint solver. *Intl. J. of Computational Geometry and Applications*, page to appear, 1994.

[5] P. Jensen. *Classical and modern mechanisms for engineers and inventors.* Marcel Dekker, 1991.

[6] R. Joan-Arinyo and A. Soto. A rule-constructive geometric constraint solver. Technical Report LSI-95-25-R, Universitat Politècnica de Catalunya, 1995.

[7] L. Lovász and M. Plummer. *Matching Theory.* North Holland, 1986.

[8] J. van Leeuven. Graph algorithms. In J. van Leeuven, editor, *Handbook of Theoretical Computer Science, Volume A*, pages 525–632. Elsevier, 1992.

[9] B.L. Van Der Waerden. *Modern Algebra (third edition).* F. Ungar Publishing Co., New York, 1950.